CONFIDENTIALITY

>> cd Alice

1. Alice creates a random key of size 128 bits and stores it in file symm.key.
>> openssl rand -out symm.key 16

2. Alice creates a file plain.txt, adds some dummy data to the file.
>> vi plain.txt

3. Alice encrypts the contents of plain.txt to cipher.txt using AES-128 algorithm in CBC mode. Use symm.key for the purpose of encryption.
>> openssl enc -aes-128-cbc -e -in plain.txt -out cipher.txt -kfile symm.key

*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

4. Alice creates a 2048 bit RSA private key. Store in file alicepriv.key.
>> openssl genrsa -out alicepriv.key -aes256 2048

Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for alicepriv.key:
Verifying - Enter pass phrase for alicepriv.key:

5. Alice extracts the public key from alicepriv.key and store in file alicepub.key.
>> openssl rsa -in alicepriv.key -out alicepub.key -pubout

Enter pass phrase for alicepriv.key:
writing RSA key

>> openssl rsa -in alicepub.key -text -noout -pubin

RSA Public-Key: (2048 bit)
Modulus:
    00:b6:05:71:95:02:bb:fb:17:1f:1d:bb:63:ea:7d:
    70:37:c5:82:b1:01:7f:50:83:80:61:0e:49:45:78:
    ac:7b:42:92:31:5d:61:d8:b5:e2:46:84:31:51:b9:
    03:81:e5:31:43:fb:9e:df:73:17:ba:5b:19:e9:82:
    6c:92:92:b5:53:aa:be:d1:1d:5b:b9:12:26:e1:21:
    6d:d1:df:fd:ea:73:cb:16:de:32:6b:26:fa:02:e0:

ba:a5:3f:5d:c1:bc:bb:c7:4d:67:d3:0d:df:e3:b8:
a2:65:c4:7b:c2:d3:bd:ee:f0:b6:cc:68:00:74:8a:
33:63:14:27:d3:45:d2:80:ff:57:0d:de:82:ec:1c:
9e:ee:c8:98:a7:1e:9c:7d:23:8b:8b:26:12:de:b5:
cd:e9:d2:7b:f0:54:e0:e9:cc:17:dc:07:32:cc:74:
6c:4c:e2:d8:b5:51:93:c7:09:08:74:1c:42:2e:57:
10:bb:3a:5f:cd:56:97:45:0c:25:18:4e:eb:61:ed:
c3:4f:43:d4:ee:a6:19:23:2f:1b:4f:68:a5:de:c8:
71:f2:e7:6f:7b:09:0b:2d:e7:40:5d:9c:88:27:f4:
3c:88:20:0e:c4:37:ca:84:3e:34:9a:2a:e2:77:a3:
b0:ab:16:82:25:ec:62:47:ca:11:c5:46:dc:21:0a:
4e:b1
Exponent: 65537 (0x10001)

6. Repeat step 4 and 5 to create private and public key of Bob. bobpriv.key and bobpub.key.

>> cd Bob
>> openssl genrsa -out bobpriv.key -aes256 2048

Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
Enter pass phrase for bobpriv.key:
Verifying - Enter pass phrase for bobpriv.key:

>> openssl rsa -in bobpriv.key -out bobpub.key -pubout

Enter pass phrase for bobpriv.key:
writing RSA key

Alice and Bob exchange their public keys.
>> cd Alice
>> cp alicepub.key ../Bob/alicepub.key

>> cd Bob
>> cp bobpub.key ../Alice/bobpub.key

7. Alice sends cipher.txt to Bob.
>> cp cipher.txt ../Bob/

8. Alice encrypts symm.key using the public key of Bob. Store in symm.enc.key.

>> cd Alice
>> openssl rsautl -in symm.key -out symm.enc.key -inkey bobpub.key -encrypt -pubin
>> cp symm.enc.key ../Bob/symm.enc.key

9. Bob decrypts symm.enc.key using his private key and stores the output in symm.dec.key.
>> cd ../Bob
>> openssl rsautl -in symm.enc.key -out symm.dec.key -inkey bobpriv.key -decrypt

Enter pass phrase for bobpriv.key:

10.Bob decrypts cipher.txt using symm.dec.key and stores the output in cipher.dec.txt. The cipher.dec.txt and plain.txt should have the same contents.
>> openssl enc -d -aes-128-cbc -in cipher.txt -out cipher.dec.txt -kfile symm.dec.key

*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.

INTEGRITY
create hash
>> openssl dgst -sha512 -binary -out hash.txt plain.txt

check hash
>> od -c hash.txt

make minor changes to plain.txt and verify hash
>> diff hash1.txt hash.txt

AUTHENTICATION
>> openssl dgst -binary -sha512 -hmac 12345 -out hash.mac plain.txt

DIGITAL SIGNATURE
>> openssl dgst -sha512 -sign alicepriv.key -out hash.sign plain.txt

Enter pass phrase for alicepriv.key:

>> openssl dgst -sha512 -verify alicepub.key -signature hash.sign plain.txt

Verified OK